Managing experimental data using FuGE

Andrew R Jones[1,*] and Allyson L. Lister[2]

[1]Department of Pre-clinical Veterinary Science, Faculty of Veterinary Science, University of Liverpool, Liverpool L69 7ZJ, UK

[2]CISBAN and School of Computing Science, Newcastle University, Newcastle upon Tyne NE1 7RU, UK

* Corresponding author: andrew.jones@liv.ac.uk

Abstract

Data management and sharing in omics science is highly challenging due to the constant evolution of experimental techniques, the range of instrument types and software used for analysis, and the high volumes of data produced. The Functional Genomics Experiment (FuGE) Model was created to provide a model for capturing descriptions of sample processing, experimental protocols and multi-dimensional data for any kind of omics experiment. FuGE has two modes of action: i) as a storage architecture for experimental workflows, and ii) as a framework for building new technology-specific data standards.

FuGE is an object model which is converted into an XML implementation for data exchange. Software toolkits have been developed for data handling and for bridging between XML data files and relational database implementations. FuGE has been adopted by the Proteomics Standards Initiative (PSI, http://www.psidev.info) for building several new data formats and it is being used in a variety of other experimental contexts, thus allowing data to be integrated across a range of experimental types to support Systems Biology approaches.

This chapter provides a practical guide for laboratories or groups wishing to manage their data, and for developers wishing to create new data formats using FuGE.

Keywords: data standards; functional genomics; data exchange; database development.

Introduction

In the proteomics domain new experimental techniques are frequently developed, presenting significant challenges for data management and sharing. Over the last ten years of proteome research, the paradigm has gradually shifted from gel electrophoresis for protein separation, to “shotgun” methods that separate complex mixtures of peptides, based around liquid chromatography. As in the early days of proteomics, the main technique for peptide and protein identification is still mass spectrometry coupled with a database search engine, yet within this space new approaches are frequently proposed for data processing and statistical analysis, several of which are outlined in other chapters. New methods have also been created for separation, including developments in multi-dimensional liquid chromatography, pre-fractionation of proteins, capillary electrophoresis, centrifugation and so on (1). In recent years, there have also been numerous methods published for quantifying proteins detected by mass spectrometry, either by relative (2, 3) or absolute measures (4, 5), and by differential gel analysis (6). As such, the types of data and metadata that must be stored are not static and the volumes of relevant data are increasing rapidly as high-throughput instruments become commonplace. In addition, few laboratories would define themselves solely as proteomics-based; most laboratories use a range of approaches to analyse their samples of interest.

In the past, databases have been created that focus on a single experimental technique, such as repositories for mass spectrometry results (7-9), microarrays (10, 11) or protein-protein interactions (12). Data management solutions are thus required which store and analyse proteome data in conjunction with results from the numerous other techniques used to interrogate samples.

One of the challenges of managing proteome data is the range of different file types produced by instruments and software, often in closed-source vendor specific formats. In recent years, several research councils and funders (e.g. BBSRC, MRC and the Wellcome Trust, for URLs see Table 1) have released data sharing policies that require omics experimental data to be made publicly available as a condition of grant funding. There are also clear benefits to all researchers if experimental data can be made publicly accessible by allowing new findings to be derived beyond the original conclusions of the study, and by facilitating improvements in analysis algorithms. The benefits are diminished, however, if data are published in vendor-specific formats, since few laboratories will have software capable of processing or analysing the files. As such, it is widely recognized that the development of standard exchange formats is important for data sharing (for example see the chapter on mzML). It is also clear that infrastructures are needed that can facilitate the rapid development of new standards and provide solutions for bringing together existing formats (both open-source and vendor-specific) in a single architecture.

The Functional Genomics Experiment (FuGE) model is a technology-independent data model for storing descriptions of experimental processes (13). FuGE contains high-level models to describe protocols, biological materials (such as samples) and multi-dimensional data. It also provides mechanisms for

referencing other resources such as ontologies, databases and external data files. FuGE can be used to describe protocols for sample processing, separation, data acquisition and data processing as well as track the flow of samples and data through the process as inputs and outputs. It can also be used as a framework for creating new data standards specific to a certain technology, such as gel electrophoresis (14). By providing a mechanism for describing the overall study, including references to data files, FuGE can integrate proteomic data with other data types, for instance those relating to genes, metabolites or phenotypes.

In this chapter we first present a brief description of the contents of the abstract FuGE Object Model (FuGE-OM) and the different concrete representations available. These include XML (FuGE-ML) for data exchange, relational database implementations and software toolkits for bridging between XML objects and database storage. The chapter then provides a practical guide for developers wishing to use FuGE either for managing experimental data or for the creation of technology-specific extensions.

Methods

FuGE is a large, complex model that has been documented in detail elsewhere (13, 15). Section therefore provides a basic introduction to the main concepts, sufficient to understand the different ways in which the model can be used. FuGE has two modes of action. First, FuGE can be used as an experimental metadata *integrator* to represent basic laboratory workflows to capture sample tracking, experimental protocols or procedures, the data files that result (represented in the relevant external format) and data processing pipelines (Section). Second, FuGE can be used as an *extensible core* to build new data formats that share a common underlying structure. In this mode, models can be created to represent

specific details about a new experimental technique for which there is no existing data standard (Section).

1.1 Overview of the FuGE model

FuGE is a model for representing the running of laboratory or computational protocols and the flow of samples and data files. There is an important distinction made between the details of the intended procedure (Protocol) and the running of the procedure (ProtocolApplication). Protocols should be defined once, with a set of sequential steps and parameters with default values. Protocols can be associated with descriptions of instruments and software, which can also have parameters. ProtocolApplications reference the Protocol that was performed, and provide the backbone of a workflow by mapping all inputs and outputs. ProtocolApplications can be annotated with the operator of the protocol, the date, any small deviations from the set protocol and any parameters that differed at runtime. This simple, flexible format is able to cover the majority of biological use cases because it abstracts the representations of experiments to the basic building blocks: procedures and their inputs and outputs. Inputs and outputs can be either materials or data. FuGE does not define detailed models for biological materials (e.g. samples) since the variety in what could be described is vast. Instead, FuGE has a structure for importing ontological annotations so they can be stored and exchanged alongside experimental descriptions. Linking materials and other similar FuGE concepts to appropriate terms in ontologies, such as those within the Open Biomedical Ontologies (OBO) Foundry (16), allows semantically rich descriptions to be created without the need for detailed FuGE models of those concepts. The OBO Foundry is a collaboration of developers of science-based ontologies created to establish a set

of principles for ontology development. Foundry ontologies are intended to be interoperable reference ontologies in the biomedical domain. For example, the Ontology for Biomedical Investigations (OBI) Consortium (http://purl.obofoundry.org/obo/obi) is developing an integrated ontology for the description of biological and clinical investigations, designed to support the consistent annotation of biomedical investigations, regardless of field of study or data type. The OBI ontology is well-suited for use within the FuGE structure, as it models a number of terms useful for describing experiments, including the protocols, instrumentation and material used, the data generated, the type of analysis performed and other artefacts generated during an investigation.

Examination of two possible uses of the ProtocolApplication entity demonstrates the differences between the integrator and extensible core modes of action of FuGE. A standard ProtocolApplication allows any number of input samples/biological materials and data files, and any number of output materials or data files. In other words, the model itself places no restrictions on what can be captured. Therefore when used as an integrator (Section), any constraints must be created by a specific software implementation or left to the user to provide sensible values representing their domain of interest. The benefit of this mode is that development time is focussed on the creation of the user interface, and not around modification of the FuGE model. However, the standard ProtocolApplication class can also be extended (Section). This can be done when using FuGE as an extensible core. For instance, a gel-scanning protocol in GelML could restrict the input to be a single two-dimensional gel, and restrict the output to be an image file. Using FuGE in this extensible core mode provides much higher control over what can be put into the model, but requires development effort in model extension. The integrator mode may be more useful for communities where the core model of FuGE is enough for their needs, or where

all constraints on experimental metadata can be easily written directly into a shared user interface. The use of FuGE as an extensible core may be more suitable to communities or groups with large numbers of complicated procedures, where providing specific extensions of FuGE concepts is the best way to ensure correct deposition of experimental metadata.

1.2 Representing laboratory workflows

For communities in which FuGE will be used in integrator mode i.e. without extension, the following general steps should be followed. For those using FuGE as an extensible core, the steps dealing with data representation and interface creation should be considered after following the stages outlined in Section 2.3.

1. *Data representation*. A suitable storage mechanism must be chosen, for example using a relational database or a native XML database. The FuGE project provides a toolkit for manipulation and validation of XML and two toolkits for relational database storage, based on different underlying technologies (http://fuge.sourceforge.net/). See Figure 1 for the roles that are played by the different technologies in the FuGE toolkits.

2. *Interface*. An interface must be created for capturing metadata about studies entered by experimentalists and to integrate the outputs from different instruments or software. Existing open-source projects such as SyMBA (http://symba.sourceforge.net) or sysFusion (from the Friedrich Miescher Institute for Biomedical Research, http://www.fmi.ch/) utilise the FuGE toolkits, providing relational database / graphical user interfaces to the core FuGE model.

3. *Storage of Protocols*. Typically a database would store experimental protocols that are commonly performed in the laboratory to be referenced by ProtocolApplications when experiments are performed.

4. *Sample tracking*. The interface should be capable of tracking the flow of samples and data files through an experimental workflow. FuGE can be used to model at the level of detail of a LIMS (Laboratory Information Management System), achieved through the use of ProtocolApplications which reference the input and outputs of each stage: samples or data files.

5. *Integration of other data standards*. In a proteomics workflow, mass spectrometry data may be represented in the mzML format, or in a vendor-specific format, such as mgf (Mascot Generic Format). Protein and peptide identifications may also be represented in the PSI's AnalysisXML format or the output produced by the search engine. The FuGE model can integrate such data files into an entire workflow by defining ProtocolApplications that reference external data files as an output. The inputs to the ProtocolApplications are descriptions of samples, for example output from a separation technology such as liquid chromatography or gel electrophoresis (Figure 2).

6. *Export of data to public repositories*. The types of metadata that should be reported about a proteomics experiment to support a publication are currently in a state of flux. As one example, the PRIDE database will support deposition of data in PSI-sanctioned formats, such as those containing MS data and protein and peptide identifications. It is expected that public repositories will support deposition of omics results tied together using FuGE-ML in the near future.

1.3 Building technology-specific extensions

FuGE can be used to develop new formats for describing particular experimental details. For proteomics, extensions have been created for representing gels (GelML) and general separations and sample processing, including liquid chromatography (spML). The PSI format for mass spectrometry database searches (peptide and protein identification) is AnalysisXML, which also makes use of several FuGE structures. Formats have also been developed using FuGE for flow cytometry, genetical genomics, RNA interference, metabolomics, microarrays and e-neuroscience which may become standards in due course. This section provides a practical guide for developers wishing to understand how FuGE can be used to create a new data format, for example to represent a particular type of experiment.

Stage 1 – Develop reporting requirements and use cases

The Proteomics Standards Initiative has developed a series of minimum reporting guidelines under the MIAPE (Minimum Information About a Proteomics Experiment) parent document. Each of the MIAPE checklists defines the minimal information that should be reported about a particular type of proteomics technique. The MIAPE parent document (17) outlines the purpose and principles of the guidelines with a series of modules, generally one per technique, including mass spectrometry (18), mass spectrometry informatics (19), gel electrophoresis (20) and others. The MIAPE documents represent a formalisation of community opinion on the types of metadata that should be captured about an experiment to allow it to be critically understood and for data to be re-analysed in the future. While some experimental techniques may not require a formal MIAPE module prior to the creation of a data exchange format, an essential first step towards developing a new format is to gain widespread input on the types of data that

laboratories wish to exchange. In many cases, this involves collecting use cases, such as different protocols used by laboratories, sets of results, references to publications and so on, and deciding which use cases should be supported by the standard.

Stage 2 –Survey existing formats and examine requirements for using FuGE

An important stage in creating new data formats is examining existing formats produced by software and instrument vendors, and those used by individual laboratories or consortia for managing their data. It is useful to identify the deficiencies of existing formats with respect to the use cases or reporting requirements that are to be supported, as identified in Stage 1.

FuGE is intended for representing metadata about experiments and representing a flow of multiple processes. For experimental techniques that encompass only a single stage producing large quantities of data, using FuGE would not be recommended, since the complexity of FuGE may slow development of the format. As described above, FuGE has a mechanism for referencing non-FuGE based formats and capturing details about the protocols used to create the data. As an example in proteomics, mzML is being developed as a stand-alone format for representing the output of a mass spectrometry. While mass spectrometers may perform two (or more) protein / peptide fragmentation stages, the data can be viewed as a discrete, atomic element, and thus there is no need for the flexible process model of FuGE. FuGE is particularly recommended for describing experimental approaches where there is significant flexibility in how different groups use the technology or where multiple different stages can be performed with outputs of one stage becoming inputs to the next.

Stage 3 – Building a FuGE extension

The FuGE UML model can be downloaded from the website (http://fuge.sourceforge.net/) and viewed with MagicDraw community edition (http://www.magicdraw.com/). The concepts identified in stage 1 should be partitioned according to the top-level FuGE classes, in particular: Protocol, ProtocolApplication, Action, Parameter, Material, InternalData and ExternalData. Classes are extended in FuGE by using a UML inheritance relationship, which is automatically applied to the XML Schema (discussed in Stage 4)

Protocol – A Protocol is intended to represent a description of an *intended* process or one that is carried out multiple times with the same settings. One example would be a standard operating procedure for sample preparation in proteomics. The Protocol class should be extended to capture specific types of parameters or steps (Action) within the protocol. For instance, in sample preparation, individual steps might be protein extraction, solubilisation and separation.

Parameter – Parameters are replaceable values within a Protocol that can be assigned a default value. The Parameter class should be extended to capture a specific type of setting within a Protocol, instrument (Equipment) or Software, such as temperature or time.

Material – Materials represent all substances used in an experiment. In practice, Material is typically used to capture details of samples of the organism being studied. The Material class has no attributes for describing its properties; instead, two mechanisms can be used to add descriptions. First, Materials can be annotated with ontology terms to describe the characteristics, for example using ontologies defined within OBI. Second, the Material class can be extended to add additional attributes. We recommend the use of ontologies particularly for describing the fundamental characteristics of the starting sample in a study,

such as species, observations, phenotypes, medical histories and so on. It is highly challenging to build data models to represent such a range of information and it is unlikely that information stored within such a model could be interpreted by any other software systems. By using ontologies, it should be possible to annotate sample descriptions with terms that are comprehensible to software systems and allow queries over repositories without requiring natural language processing or synonym searches. We recommend extending the Material class in FuGE to store details about substances related directly to the experimental technique, for example those requested by a reporting guideline document. As an example, in GelML, the Material class has been extended to describe electrophoretic gels. The Gel class has attributes for capturing the physical dimensions, the percentage acrylamide in the gel, and the ratio of acrylamide to a crosslinking agent. Such specific details are required by the MIAPE document for gel electrophoresis (20), and as such they are included in the data model as attributes on the Gel class to simplify the capture of these values.

Data – FuGE has a structure which can be used to describe multi-dimensional data (InternalData). The model first defines the dimensions of data, and the values are stored separately in matrices. The data matrices can be accessed using coordinates defined by a combination of the dimensions. The FuGE data model should be extended to describe multi-dimensional data with a regular size of dimensions, for instance this would be appropriate for describing protein arrays. In other cases, individual elements can be created in FuGE to describe simple results (for example as input/output parameter values), or an external data format can be referenced as an input or output from a ProtocolApplication, such as spectra captured as ExternalData.

ProtocolApplication - ProtocolApplication represents the running of a Protocol and maps the inputs and outputs of the processes. ProtocolApplication should be extended when a specific type of Protocol is being run, and when it is necessary to define specific types of samples (Material) or data files as inputs or outputs.

Stage 4 – Creating an XML data interchange format

The object model is converted to an XML Schema using the XSD STK (http://fuge.sourceforge.net/stks/xsd-stk/), which defines and constrains what can be represented in XML. It is a lightweight toolkit designed expressly for those wishing to either manipulate the FuGE XSD (XML Schema Definition) or generate a new FuGE-based XSD for a particular purpose. The rules used to convert the object model into an XML Schema are described in the FuGE specification document (15). Where an extension has been built to describe a particular technology, the same toolkit can be used to create a new XSD for the extended model.

Summary

FuGE has been created to simplify the development of data management solutions for life sciences and attempts to unify data formats created for different technologies. The chapter has provided a brief guide in using FuGE for either of its intended purposes, as a metadata integrator and as an extensible core for creating new data formats. For proteomics data management, both mechanisms are in current use and a community of developers continue to contribute tools and new extensions, which can be accessed via the FuGE website.

References

Figures

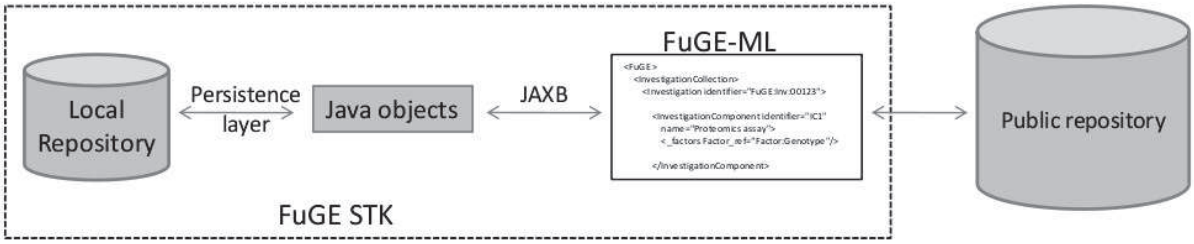

Figure 1 The interplay between different technologies for manipulating experimental descriptions within FuGE. A local repository can be implemented using a relational database. A Persistence layer is used to convert between in-memory objects (e.g. in Java) and database storage. The FuGE project provides two toolkits employing different persistence layers, based on EJB (http://java.sun.com/products/ejb/) and Hibernate (http://www.hibernate.org/). JAXB (https://jaxb.dev.java.net/) mappings are included in the toolkit which can convert in-memory Java objects to an XML representation, FuGE-ML, which will allow data to be sent to public repositories that accept FuGE-formatted data.

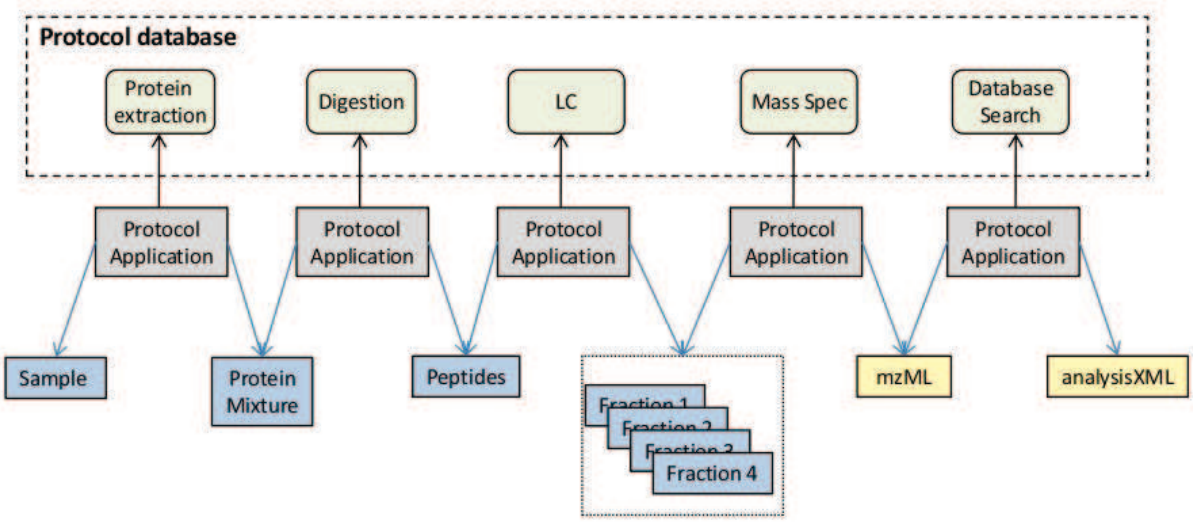

Figure 2 An example LC-MS experiment represented in FuGE. The inputs and outputs to each ProtocolApplication are either descriptions of samples (annotated with ontology terms) or data files, such as mzML or analysisXML.

Organisation	URL
Wellcome Trust	http://www.wellcome.ac.uk/About-us/Policy/Policy-and-position-statements/WTX035043.htm
BBSRC	http://www.bbsrc.ac.uk/publications/policy/data_sharing_policy.pdf
MRC	http://www.mrc.ac.uk/Ourresearch/Ethicsresearchguidance/Datasharinginitiative/index.htm

Table 1 Data sharing policies of the main funders of biological research.